Nicht für den Prüfling bestimmt

I Erläuterungen

Voraussetzungen gemäß KCBG und Abiturerlassen BG jeweils in der für den Abiturjahrgang geltenden Fassung

Standardbezug
Die nachfolgend ausgewiesenen Kompetenzbereiche sind für die Bearbeitung der jeweiligen Aufgabe besonders bedeutsam. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Kompetenzen für die Bearbeitung der Aufgabe nachrangig bedeutsam sein, zumal die Kompetenzen in engem Bezug zueinander stehen. Die Operationalisierung des Bezugs zu den Kompetenzbereichen des Standardbezugs erfolgt in Abschnitt II.

Aufgabe	Kompetenzen				
	K1	K2	K3	K4	K5
1.1			X		
1.2			X		
1.3			X		
1.4				X	
2.1	X				
2.2	X	X			
2.3		X			
2.4	X				
2.5		X			
3.1					X
3.2			X		
3.3				X	
3.4.1		X			
3.4.2					X

Inhaltlicher Bezug
Die nachfolgend ausgewiesenen Themenfelder sind die wesentliche inhaltliche Grundlage für die vorliegenden Aufgaben. Darüber hinaus können weitere, hier nicht explizit ausgewiesene Themenfelder für die Bearbeitung nachrangig bedeutsam sein.
Q1: Objektorientierte Softwareentwicklung
Q2: Digitale Steuerungstechnik
Q3: Prozessautomatisierung
verbindliche Themenfelder: Objektmodellierung (Q1.1), Implementierung von Klassen und ihren Beziehungen (Q1.2), Synthese statischer und sequentieller Logikschaltungen (Q2.1), Mikrocontroller (Q2.2), Einführung in die Prozessautomatisierung (Q3.1)

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Nicht für den Prüfling bestimmt

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
1.1	entwickeln, bestimmen Entwickler neue Algorithmen erstellen optimieren vorhandener Algorithmen Simulator Simulationen entwerfen <<include>> Rasenmähroboter anlegen <<include>> Rasenfläche anlegen <<extend>> Condition: {Simulation ist noch nicht vorhanden} extension point: Entwurf nötig Simulationen durchführen extension points Entwurf nötig <<extend>> Algorithmus importieren Simulationen löschen			
	entwickeln		5	
	bestimmen	5		

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.2	ergänzen, modellieren <<abstract>> *Algorithmus* - name : String - ersteller : String - derzeitImEinsatz : boolean - idAlgo : int - algorithmus : String + algorithmusExportieren() : void + fahreVorwärts(zeitS : int) : void + fahreRückwärts(zeitS : int) : void + dreheLinks(winkel : int) : void + dreheRechts(winkel : int) : void + pruefeHindernisVorne() : boolean + zufallswinkel() : int IterativerAlgorithmus - anzahlSchritte : int RekursiverAlgorithmus - rekursionstiefe : int Simulation - name : String - ersteller : String - datum : String + Simulation(n : String, e : String, d : String) + simulationDurchführen() : void + simulationEntwerfen() : void + simulationLöschen() : void - rasenmähroboterAnlegen(n : String, t : int, a : int, m : int) : void - rasenflächeAnlegen(n : String, l : int, b : int, e : boolean) : void + neuenAlgorithmusErstellen() : void + algorithmusLaden() : void Rasenfläche - name : String - laenge : int - breite : int - eingefasst : boolean + Rasenfläche(n : String, l : int, b : int, e : boolean) + berechneFlaeche() : int Rasenmähroboter - name : String - typennummer : int - akkustand : int - messeranzahl : int + Rasenmähroboter(n : String, t : int, a : int, m : int) + pruefeAkkustand() : int + berechneReichweite(akkustand : int) : int GUI verwaltet * 1..* 1 1..* 1 1..*			
	ergänzen	3		
	modellieren		2	5

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.3	überführen (Objektdiagramm, siehe unten)	7		
1.4	implementieren (Code, siehe unten)		9	4

1.3 überführen

epsilonP4 : Simulation
name = "EpsilonP4"
ersteller = "Edwin B. Budding"
datum = 05.02.2021

easyCutter42 : Rasenmähroboter
name = "EasyCutter42"
typennummer = 1337
akkustand = 100
messeranzahl = 3

rasenfläche1 : Rasenfläche
name = "Rasenfläche1"
laenge = 10
breite = 55
eingefasst = true

rasenfläche2 : Rasenfläche
name = "Rasenfläche2"
laenge = 4
breite = 9
eingefasst = false

1.4 implementieren

```
//Klasse Simulation

import java.util.ArrayList;

public class Simulation {

private String name;
private String ersteller;
private String datum;

private ArrayList<Rasenmähroboter> mäherListe;
private ArrayList<Rasenfläche> flächenListe;


public Simulation(String n, String e, String d) {
      this.name = n;
      this.ersteller = e;
      this.datum = d;

      //Erzeugen der ArrayLists
      mäherListe = new ArrayList<Rasenmähroboter>();
      flächenListe  = new ArrayList<Rasenfläche>();
      //Umsetzung der Komposition
      rasenmähroboterAnlegen("Freelexo", 600, 100, 3);
      rasenflächeAnlegen("Rasenfläche1", 10, 55, true);
}

//Nutzung der zur Erzeugung der Objekte vorgesehenen Metho-
den
private void rasenmähroboterAnlegen(String n, int t, int a,
int m) {
      Rasenmähroboter rmr = new Rasenmähroboter(n, t, a,
m);
      mäherListe.add(rmr);
}
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III
	(Code siehe unten)			
	Summe 40	15	16	9

```
private void rasenflächeAnlegen(String n, int l, int b,
boolean e) {
      Rasenfläche rf = new Rasenfläche(n, l,b, e);
      flächenListe.add(rf);
}
}

//Klasse Rasenmähroboter

public class Rasenmähroboter {

      private String name;
      private int typennummer;
      private int akkustand;
      private int messeranzahl;

      public Rasenmähroboter(String n, int t, int a, int m)
{
            this.name = n;
            this.typennummer = t;
            this.akkustand = a;
            this.messeranzahl = m;
      }

      public int pruefeAkkustand() {
            return 0;
      }

      public int berechneReichweite(int a) {
            return 0;
      }
}

//Klasse Rasenfläche

public class Rasenfläche {

      private String name;
      private int laenge;
      private int breite;
      private boolean eingefasst;

      public Rasenfläche(String n, int l, int b, boolean e)
{
            this.name = n;
            this.laenge = l;
            this.breite = b;
            this.eingefasst = e;
      }

      public int berechneFlaeche(int a) {
            return 0;
      }
}
```

Hinweis: Die grau hinterlegten Zeilen sind nicht Bestandteil der Aufgabe.

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
2.1	analysieren, beschreiben Das Bauteil „MC74HC112A“ ist ein complementary metal-oxide-semiconductor-Bauteil, kurz CMOS-Bauteil, welches zwei JK-Flipflops mit Setz- und Rücksetzeingang enthält. Die Flipflops sind taktflankengesteuert und reagieren auf die fallende Flanke. Setz- und Rücksetzeingänge sind negiert zu beschalten. Neben den Anschlüssen für die Flipflops verfügt das Bauteil noch über V_{CC}- und GND-Eingänge. Der function table gibt Aufschluss über das Verhalten eines JK-Flipflops bei gleichzeitig anliegender negativer Flanke. Sind beide Eingänge low, ändern sich die Werte an den Ausgängen nicht. Sind beide Eingänge high, schaltet das Flipflop in den Toogle-Modus, bei dem zu jedem Takt die Werte an den Ausgängen zwischen high und low wechseln. Ist der J-Eingang low und der K-Eingang high, liegt am Ausgang Q ein low-Signal an. Ist der J-Eingang high und der K-Eingang low, liegt am Ausgang Q ein high-Signal an. analysieren beschreiben	 2	2	2
2.2	dokumentieren Das charakteristische Verhalten eines D-Flipflops ist, dass immer, wenn der Taktimpuls anliegt, der Wert vom Eingang am Ausgang übernommen wird. Ein JK-Flipflop lässt sich einfach zu einem D-Flipflop verschalten. Dazu erhält ein Eingang, J oder K, den negierten Eingangswert des jeweils anderen Eingangs. Dem Verhalten nach ist dann folglich der Ausgang Q einen Takt später exakt der Wert, den man vorher in das Flipflop eingegeben hat.		3	2

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
2.3	entwerfen		3	4
2.4	nennen Vier Takte, nachdem der erste Wert des auszulesenden Datenstroms am Eingang des ersten D-Flipflops angelegt wurde, kann dieser am Ausgang Q4 wieder abgegriffen werden.	2		

Nicht für den Prüfling bestimmt

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.5	vereinfachen (siehe unten: q_{2n+1}, q_{1n+1}, q_{0n+1})		4	2
	Summe 26	**4**	**12**	**10**

vereinfachen

q_{2n+1}:

		q_0	q_0	
	0	0	X	0
q_1	0	1	X	X
			q_2	q_2

$$q_{2n+1} = (q_0 \wedge q_1)$$

q_{1n+1}:

		q_0	q_0	
	0	1	X	0
q_1	1	0	X	X
			q_2	q_2

$$q_{1n+1} = (q_0 \wedge \overline{q_1}) \vee (\overline{q_0} \wedge q_1)$$

q_{0n+1}:

		q_0	q_0	
	1	0	X	0
q_1	1	0	X	X
			q_2	q_2

$$q_{0n+1} = (\overline{q_0} \wedge \overline{q_2})$$

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
3.1	erläutern, zeichnen Für den Intel 8086 Mikroprozessor muss jeweils ein Port als Eingang und ein Port als Ausgang definiert werden. Aus diesem Grund ist die vorgegebene Pin-belegung auf zwei Ports aufgeteilt worden: Port 10 in, Port 20 out. MC Port 10 (in) Port 20 (out) Pin 6: Reinigungsmodus Pin 0: Mähwerk ein Pin 2: Rechtslauf Pin 1: Linkslauf erläutern zeichnen	 2 2		

Aufg.	erwartete Leistungen	BE I	BE II	BE III
3.2	darstellen endlos Mähwerk ein? (Pin 0 = high) Ja: Reinigungsmodus? (Pin 6 = high) Ja: Mähwerk noch an? (Pin 0 = high) Ja: Rechtslauf deaktivieren (Pin 2 = low) 10 Sekunden warten Nein: ø Linkslauf aktivieren (Pin 1 = high) 15 Sekunden warten Linkslauf deaktivieren (Pin 1 = low) 10 Sekunden warten Nein: Rechtslauf aktivieren (Pin 2 = high) Nein: Rechtslauf deaktivieren (Pin 2 = low) Linkslauf deaktivieren (Pin 1 = low)	4	6	
3.3	implementieren ; Portbelegung Port 10 (in) / Port 20 (out): org 100h endlos: ;Endlosschleife IN AL, 10 ;Port 10 einlesen SHR AL, 0 ;Rightshift mit 0		3	7

Aufg.	**erwartete Leistungen**	**BE**		
		I	**II**	**III**
	<pre>AND AL, 00000001b CMP AL, 1 ;Pin 0 = 1? JNE aus ;Wenn 0, dann ausschalten JMP starten ;Wenn 1, dann starten aus: MOV AL, 00000000b OUT 20, AL ;Motor ausschalten an Port 20 JMP endlos starten: IN AL, 10 ;Port 10 einlesen SHR AL, 6 ;Rightshift mit 6 AND AL, 00000001b CMP AL, 1 ;Pin 6 = 1? JE reinigungsmodus ;Wenn 1, dann Reinigung MOV AL, 00000100b ;normaler Mähbetrieb an OUT 20, AL JMP endlos reinigungsmodus: IN AL, 10 ;Maehwerk noch an? SHR AL, 0 AND AL, 00000001b CMP AL, 1 ;Pin 0 = 1? JNE reinigen ;Wenn Maehwerk aus, dann Reinigung starten MOV AL, 00000000b ;ansonsten bremsen -> Motor aus OUT 20, AL CALL WAIT_10s reinigen: MOV AL, 00000010b ;Linkslauf an OUT 20, AL CALL WAIT_15s MOV AL, 00000000b ;Motor aus OUT 20, AL CALL WAIT_10s JMP endlos RET</pre>			
3.4.1	erläutern Ein kapazitiver Sensor funktioniert grundsätzlich nach dem Prinzip eines elektrischen Kondensators. Das Konzept basiert darauf, dass zwei Elektroden, eine davon kann die zu messende Oberfläche sein, die „Platten“ eines elektrischen Kondensators bilden. Dessen Kapazität bzw. Kapazitätsänderung wird gemessen.	3	1	

Aufg.	**erwartete Leistungen**	**BE I**	**BE II**	**BE III**
	Dabei kann die Kapazitätsänderung auf verschiedene Arten herbeigeführt werden: – Eine Platte wird durch den zu messenden Effekt verschoben. – Die Platte wird verformt, wodurch sich der Plattenabstand und damit die elektrisch messbare Kapazität ändert. – Die Platten sind starr und die Kapazität ändert sich, weil entweder elektrisch leitendes Material oder ein Dielektrikum in die unmittelbare Umgebung gebracht wird (z. B. zwischen die Platten). – Die wirksame Plattenfläche ändert sich, indem die Platten wie bei einem Drehkondensator gegeneinander verschoben werden. – Der zu messende Effekt beeinflusst die Permittivität (Dielektrizitätskonstante) des Dielektrikums. Die Kapazitätsänderung wird dann erfasst und in Relation zur Ausgangsgröße gesetzt. Somit lassen sich Rückschlüsse auf den zu untersuchenden Effekt feststellen.			
3.4.2	entwickeln Es ist generell möglich einen kapazitiven Sensor zur Feststellung, ob sich der Rasenmähroboter auf der Ladestation befindet, zu benutzen. Dazu könnte man einen kapazitiven Drucksensor verwenden. Dieser müsste am Boden der Ladestation unter einer beweglichen Platte installiert sein, die beim Befahren durch den Roboter den Druck auf die Membran des kapazitiven Drucksensors erhöht. So würde durch das Gewicht des auffahrenden Rasenmähroboters die Membrane, die als eine Kondensatorplatte fungiert, eingedrückt und der Abstand der beiden Elektronen zueinander geändert. Damit ändert sich die Kapazität, welche dann ausgewertet werden könnte.			4
	skizzieren Drucksensor Membran (Platte 1) Platte 2		2	
	Summe 34	**11**	**12**	**11**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung. Bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache sind die Bestimmungen des § 9 Abs. 12 Satz 3 OAVO in Verbindung mit Anlage 9b anzuwenden.

Bei der Bewertung und Beurteilung der Übersetzungsleistung in den Fächern Latein und Altgriechisch sind die Bestimmungen des § 9 Abs. 14 OAVO in Verbindung mit Anlage 9c anzuwenden.

Der Fehlerindex ist nach Anlage 9b zu § 9 Abs. 12 OAVO zu berechnen. Für die Ermittlung der Punkte nach Anlage 9a zu § 9 Abs. 12 OAVO sowie Anlage 9c zu § 9 Abs. 14 OAVO wird jeweils der ganzzahlige nicht gerundete Prozentsatz bzw. Fehlerindex zugrunde gelegt.

Für die Bewertung in den modernen Fremdsprachen ist der „Erlass zur Bewertung und Beurteilung von schriftlichen Arbeiten in allen Grund- und Leistungskursen der neu beginnenden und fortgeführten modernen Fremdsprachen in der gymnasialen Oberstufe, dem beruflichen Gymnasium, dem Abendgymnasium und dem Hessenkolleg“ vom 7. August 2020 (ABl. S. 519) zugrunde zu legen. Demnach erfolgt die Bewertung und Beurteilung mit der Maßgabe, dass lediglich bei der Ermittlung des Prüfungsergebnisses (Note) aus Prüfungsteil 1 und 2 gerundet wird.

Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen (Abiturerlass)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Als Kriterien für die Bewertung und Beurteilung dienen unter Beachtung der Zielsetzung der gymnasialen Oberstufe nach § 1 Abs. 2 OAVO neben dem Inhaltlichen auch die in den Kerncurricula genannten überfachlichen Kompetenzen, insbesondere die Sprachkompetenz und Wissenschaftspropädeutik; dies zeigt sich u.a. in qualitativen Merkmalen wie Strukturierung, Differenziertheit, (fach-)sprachlicher Gestaltung und Schlüssigkeit der Argumentation.

Im Fach Technische Informatik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass mindestens 45% der zu vergebenden BE erreicht werden. Ein Prüfungsergebnis von **11 Punkten (gut)** setzt voraus, dass mindestens 75% der zu vergebenden BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	**Bewertungseinheiten in den Anforderungsbereichen**			**Summe**
	AFB I	**AFB II**	**AFB III**	
1	15	16	9	**40**
2	4	12	10	**26**
3	11	12	11	**34**
Summe	**30**	**40**	**30**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.